TOOLS FOR USER MODIFICAITON OF OPTIMAL ROADMAPS

An Undergraduate Research Scholars Thesis

by

NICOLE VIOLET JULIAN

Submitted to Honors and Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Research Advisor: Nancy M. Amato

May 2014

Major: Computer Science

TABLE OF CONTENTS

Page

ABSTRACT

Tools for User Modificaiton of Optimal Roadmaps . (May 2014)

Nicole Violet Julian
Department of Computer Science
Department of no
Texas A&M University

Research Advisor: Dr. Nancy M. Amato
Department of Computer Science

Robotic motion planning is a ubiquitous field of study, with innumerable applications in science, engineering, and beyond. At its core, however, motion planning is infeasible for many complex problems. Sampling-based algorithms address this issue by building an approximate model of the planning space, while optimal planners extend this to provide desirable guarantees on solution features (e.g., shortest paths). Unfortunately, these guarantees can require the creation of dense, cumbersome planning graphs. Automatic refinement algorithms can help to sparsify these dense graphs, though they may be costly themselves if they affect the quality of the original solution. In another direction, harnessing human intuition with user-guided planning strategies has also shown promise. In this research, we seek to combine the unique strengths of human and machine reasoning with a foundational, interactive toolset for graph modification and, thus, to overcome some weaknesses inherent in either alone. We provide a visual interface that allows the user to modify a pre-computed planning graph by adding, removing, and adjusting vertices and edges as desired, with reciprocal feedback from the planner on the feasibility of each operation. This provides a more adaptable way to improve graph quality–e.g., by sparsifying particular areas based on the unique dynamics of the environment, which are easily and naturally conceptualized by human instinct. In experiments, we found our tools to be quite helpful in improving some measures of graph quality, while their benefits on others dependended on the intuitiveness of the user interface.

DEDICATION

This work is dedicated to Carmen and Louis, who are, in many ways, my *older* siblings.

ACKNOWLEDGMENTS

I would like to acknowledge my advisor, Dr. Nancy Amato, for her encouragement and guidance throughout the development of this thesis–and all thoughout my computer science career.

I would also like to acknowledge Jory Denny, my graduate student mentor, for keeping me on track and showing me that I am capable of more than I realize.

A final acknowledgement goes to Tammis Sherman, Dr. Duncan MacKenzie, and Plamen Ivanov for their support, patience, and feedback throughout the Undergraduate Research Scholars Program.

CHAPTER I

INTRODUCTION

Motion planning is the problem of finding a valid path that takes a movable object from a start position to a goal position within an environment. It is a central element of robotics and has important applications in other areas such as bioinformatics [24], automated assembly [2], and gaming/virtual reality [18]. Unfortunately, exact motion planning is computationally intractable [22] for all but the simplest systems. In other words, as the complexity of a robot increases, the cost of deterministically finding a solution increases exponentially. Motion planning research has addressed this issue in two important ways: by developing and refining automatic algorithms, and by exploring user-guided systems. Considered separately, both of these approaches have compelling strengths and weaknesses.

On the fully automatic side, sampling-based methods such as the Probabilistic Roadmap Method (PRM) [15] and Rapidly-exploring Random Tree (RRT) [16] have arisen. These avoid explicit representations of high-dimensional spaces and instead rely on approximate *roadmaps* (graphs) of the environment for practicality and efficiency. Though more intelligent variants [1, 4, 29] have developed over time, traditional sampling-based planners are not always optimal in terms of cost metrics such as path length and execution time. These issues have accordingly been addressed by optimal sampling-based motion planning methods such as those presented in [14]. Though these algorithms can provide desirable guarantees on solution features, they do so at the expense of creating prohibitively large roadmaps. Methods such as [20] have helped to sparsify these dense graphs, with further refinements in computational speed [28], memory requirements [6], and the sparsification process itself [7, 23] arising soon after.

Despite the appeal of clever automatic planners, some situations, such as tricky narrow passages, are simply too difficult for a machine to handle alone. Thus, on the more manual side are user-guided planning strategies such as [10, 2, 13, 8, 21]. These methods capitalize

on the operator's intuition–e.g., for finding a trajectory through a narrow passage, which is notoriously difficult for automatic planners yet often trivial to the human eye.

Fully automatic and manually-influenced planners each suffer from unique weaknesses. In the automatic case, there are often tradeoffs between important factors such as memory requirements and graph sparsity. Furthermore, the customizability of automatic planners is very limited. Many qualitative aspects (e.g., selecting good input parameters) require broad, environmental considerations that a computer alone cannot capture. One-way user-guidance systems are likewise problematic. Though a user may, for example, provide a series of waypoints to help a planner navigate a narrow passage, these waypoints may not always be as useful and accurate as desired, especially in the case of a non-expert operator [12]. The human mind, though appealing for its high-level, instinctive nature, cannot offer the exactness and consistency of a machine, which can be crucial for certain requirements such as strict obstacle clearance or low-level tasks such as collision checking.

The described disadvantages of both automatic and manual planners can, in many ways, be attributed to the unidirectional natures of such systems. A computer alone may miss critical parts of the "big picture" in a planning problem. Likewise, providing feedback only from an operator to the system is not always ideal for the finer points and leaves room for human error. It is not enough for the user to guide the planner–to overcome the weaknesses inherent in both, the planner and user should help *one another*. In this work, we take a preliminary look at this cooperative ideal by providing tools for user manipulation of the planning graph itself.

More specifically, the planning graph as is conceptualized as $G = (V, E)$. V represents the set of *vertices*–namely, the feasible robot configurations that have been sampled. E represents the set of *edges*, or the feasible transitions that have been validated between these samples. We allow users to refine and customize G with tools for *motion editing*, which we define as the lowest level of interaction between a human and a planner. These tools allow for the following graph modifications:

- Vertex/Edge Insertion
- Vertex/Edge Removal
- Vertex Adjustment
- Edge Adjustment
- Vertex Merging

This allows a person to directly–and far more flexibly–perform many of the map quality improvements brought about by automatic methods such as [14] and [20]. Of critical importance, as well, are the responses that these tools give back as they are used, such as inverting the color of a vertex that is placed in an infeasible area. In this way, the benefits of two-way interaction are truly realized. As the human provides valuable instinct to the post-processing steps, the planner reciprocates with its own strengths (e.g., enforcing fine constraints) so that the instinct is applied as effectively as possible. More concisely, the contributions of this work include:

- Introducing a simple toolset for *motion editing*, whereby a user can modify a pre-computed motion planning graph, with reciprocal validity feedback from the planner.
- Evaluating the effects that this toolset has on graph quality in various 2D scenarios, as compared to an automatic refinement algorithm.

In experiments, we found the motion editing tools to be quite helpful in improving some measures of graph quality, e.g., by enabling large vertex and edge density reductions. For other measures, the benefits were largely dependent on the intuitiveness of the user interface.

CHAPTER II

PRELIMINARIES AND RELATED WORK

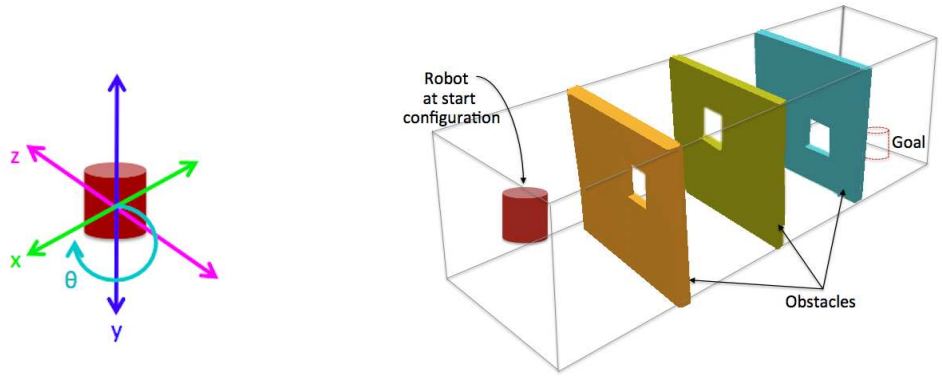
Preliminaries

We define a *robot* (Figure II.1a) as a movable object whose position can be described by n *degrees of freedom* (DOFs). A degree of freedom is a dimension of movement; e.g., each of the x, y, and z coordinates in 3-D space or the angle of a joint. Thus, a robot's overall configuration can be uniquely described as a point $\langle x_1, x_2, ..., x_n \rangle$ in n-dimensional space, where x_i is the ith DOF. The *configuration space* [22], or C_{space}, consists of all possible robot configurations, while the *free space*, C_{free}, is the subset of C_{space} that represents all feasible configurations (e.g., those not in collision with obstacles or the robot itself). The union of infeasible configurations is known as the *obstacle space*, or C_{obst}. We thus define the *motion planning problem* (Figure II.1b) as the problem of finding a continuous trajectory in C_{free} that takes a robot from a starting configuration to a goal configuration. Finding an exact solution in C_{free} is generally infeasible, as it requires the explicit computation of C_{obst} boundaries. On the other hand, determining the validity of a configuration is much easier and can be done quite efficiently, e.g., with a collision detection test in the *workspace*, the robot's natural space.

Related Work

Sampling-Based Motion Planners

Sampling-based methods exploit the ease of validating configurations in the workspace to create an approximate mapping of C_{free}. Two paradigms of sampling-based planning are the Probabilistic Roadmap Method (PRM) [15] and the Rapidly-Exploring Random Tree (RRT) [16]. In particular, PRM builds a roadmap (graph) of feasible configurations, which provides important information about their connectivity.



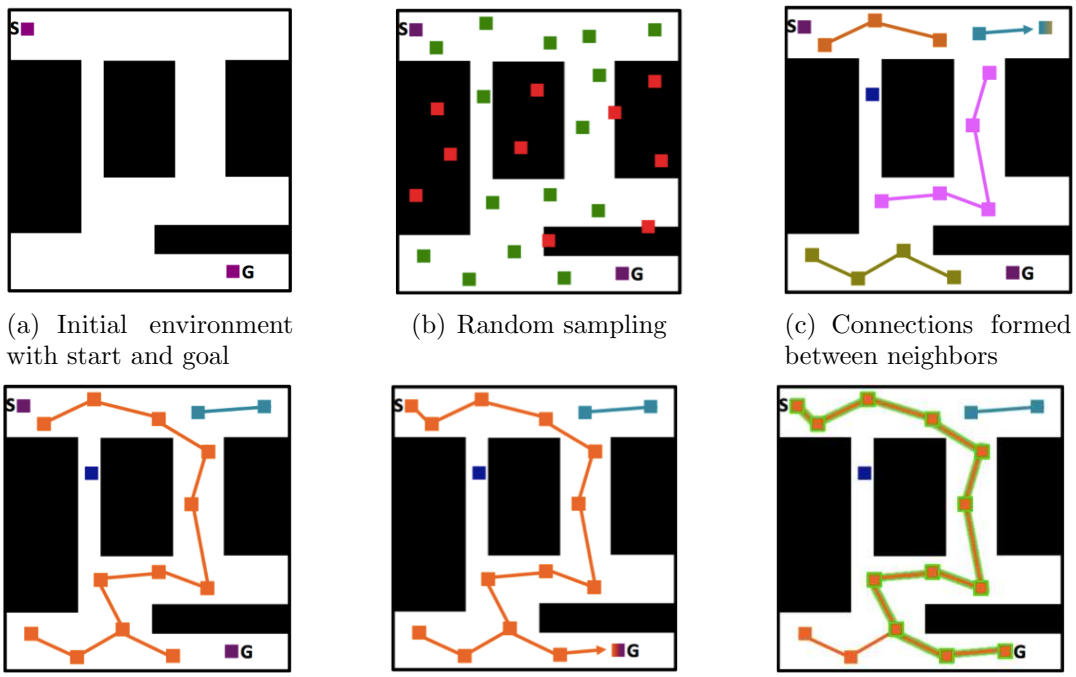
(a) A 4-DOF robot (b) A planning problem

Fig. II.1.: Motion planning basics

In the first phase, random samples are generated in the C_{space}, and those that are valid (e.g., collision-free) are kept as nodes for the roadmap (Figure II.2a-b). Next, neighboring nodes are selected as candidates for connection via a suitable distance metric, such as straight-line distance. The connections are then attempted by local planners and represented as roadmap edges if successful (Figure II.2c-d). Finally, a start and goal configuration can be connected to the roadmap (Figure II.2e), which is then queried for a path (Figure II.2f) with a graph search method, e.g., A*. Initial PRMs were successful in solving many problems far from the reach of traditional, deterministic methods, and they were followed by a number of variants, such as [1, 4, 29, 9, 5, 31], that addressed important challenges such as mapping configurations in narrow passages.

Optimality in Sampling-Based Motion Planning

In [14], Karaman and Frazzoli prove the asymptotic non-optimality of PRM and RRT and introduce PRM*, RRG, and RRT* to address this limitation. These methods only attempt connections to a node's optimal neighbors, the number of which is proportional to $log(n)$, where n is the total number of vertices in the roadmap. This allows for asymptotically optimal paths; e.g., paths guaranteed to be the shortest possible as the number of samples approaches infinity. Though this quality is desirable, the large size and high density of



(a) Initial environment with start and goal

(b) Random sampling

(c) Connections formed between neighbors

(d) Connections further along

(e) Start and goal connection

(f) Query for path

Fig. II.2.: Basic PRM algorithm

roadmaps constructed with these methods makes them impractical for efficient, anytime planning.

In [20], graph spanners are used to derive sparser, asymptotically near-optimal (i.e., no more than a constant factor worse than optimal) roadmaps with similar path quality to those produced in [14]. In one approach, a sequential algorithm removes redundant edges from pre-constructed, dense maps. This leads to significant space savings and improved query resolution time, though the length of resulting paths increases as more edges are removed. A more expensive method, the incremental roadmap spanner (IRS), interleaves with roadmap construction and rejects unnecessary edges before collision checks are performed. This likewise produces sparse roadmaps with faster querying times, and a simple smoothing method can be applied to bring path quality closer to that of denser graphs. The computational

speed of IRS is improved to be essentially constant per vertex in [28], which uses a streaming spanner algorithm to prune PRM* roadmaps.

Despite their benefits, graph spanners only remove edges and can still result in large roadmaps. In [7], the SPARS algorithm is introduced, which provides asymptotic optimality and ensures that the probability of adding new *nodes* to a roadmap converges to zero. SPARS is modified to reduce memory costs in [6], which provides near-optimal guarantees without the need for storing a dense graph. Redundant vertex removal is similarly exploited in [23], which uses edge contractions to construct a more compact representation of a pre-computed, PRM-type input graph. It is important to note that this strategy–or, more broadly, anything fully automatic–is essentially realized by a generalized, global rule set. This, by nature, limits the customizability and adaptability of such solutions.

User-Guided Motion Planning

Previous work incorporating human influence with motion planning has generally fallen into two categories. In the first approach, the human performs global scene analysis of the workspace, while the machine is responsible for jobs that require more accuracy, such as collision detection and path extraction. In [10, 2], the user can select critical configurations in interesting regions, while the planner handles the rest of the sampling, collision checking, and pathfinding between subgoals. In [11], the user is responsible for controlling an arm's linkage, while the machine takes care of the wrist. In [17], when the operator uses a haptic probe to designate the desired speed and the rate of turn for the robot, the machine performs close-range obstacle avoidance and provides force feedback to the operator.

In the second approach to human-assisted planning, Ivanisevic and Lumelsky [13, 12] investigate the idea of converting workspace into C_{space}. This allows the robot to be represented as a point, which is much easier for a human to visualize and control. Other relevant studies have exploited situations where planners typically perform poorly, such as dynamic environments. In [8], the human can intervene to handle events such as unexpected obstacles; when the human operation finishes, the machine can resume control without any replanning. In

[21], the deliberative planner uses a potential field to obtain a set of configurations for the robot, while reactive behaviors handle the dynamic obstacles that may appear as the robot moves toward the target. User inputs are combined with these two techniques to allow the user maximum control while preventing collisions. It is important to note that in this and the previously described works, a strong paradigm of two-way feedback is missing.

A few truly cooperative planning approaches have indeed been explored. An early example is [3], in which the automatic planner utilizes haptically-generated user input and, in return, provides reactive force feedback and communicates its progress through a visual overlay of the scene. In [25], Taïx et. al. present Interactive RRT (I-RRT), in which the user controls an avatar of the robot in a virtual representation of the workspace. The algorithm biases sampling near the avatar's position while providing feedback to the user through a haptic device and/or node coloring. In this way, simultaneous cooperation is achieved without the two-stage decoupling of user and planner contributions common to other approaches. In [30], Yan et. al. similarly explore collaborative planning through user manipulation of a force feedback device. They incorporate relaxation of collision constraints, as well as the random retraction of rough paths and their connection via a bidirectional RRT. Though the schemes of [25] and [30] are promising, they are limited to RRT approaches and require continuous user input throughout the planning process; in fact, I-RRT alone has no completeness or optimality guarantees. More important, however, is that no attempts have been made to lay a comprehensive groundwork for cooperative planning as a whole.

CHAPTER III

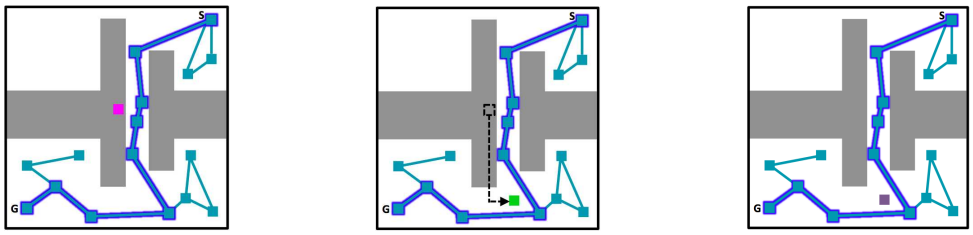
TOOLS FOR MOTION EDITING

We define *collaborative motion planning* as a paradigm of two-way interaction that integrates the advantages of human cognition and computer reasoning. This is achieved through a reciprocal feedback exchange that is, on the whole, more beneficial than the sum of its unidirectional parts. In one direction, the user aids the planner by manipulating objects in the system or providing directions; e.g., by specifying tasks or even manipulating the roadmap itself. In the other direction, the planner aids the user; e.g., through visual cues or, in the case of an online exchange, attraction/repulsion to certain trajectories or regions in the environment.

This work focuses on *motion editing*, the most basic level of collaborative planning. When higher levels of cooperation are not enough, motion editing allows a user to interact directly with the planning graph itself. In a narrow passage, for example, the user could directly input a key configuration, which could help the planner overcome a connection bottleneck much more efficiently.

Map Editing Utilities

The motion editing tools presented in this work provide for offline, collaborative roadmap manipulation and aim to lay a strong groundwork for natural, practical, and efficient real-time techniques. Recall that the map is formalized as a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. We can further consider G as a set of one or more *connected components*; i.e., subgraphs in which any two vertices are connected by a path. This work extends the visualization tools in [27] to provide simple interfaces for the following operations:

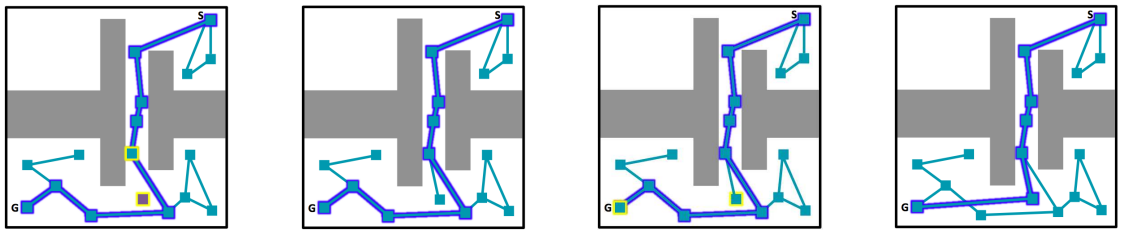


(a) Initial vertex insertion at origin; magenta = invalid

(b) Adjustment to desired (and valid) configuration

(c) New vertex confirmed

Fig. III.1.: Use of the vertex insertion tool. In this and subsequent examples, the shortest available path is outlined in blue.



(a) Selection of endpoints for new edge

(b) New edge confirmed

(c) Selection of endpoints for another edge

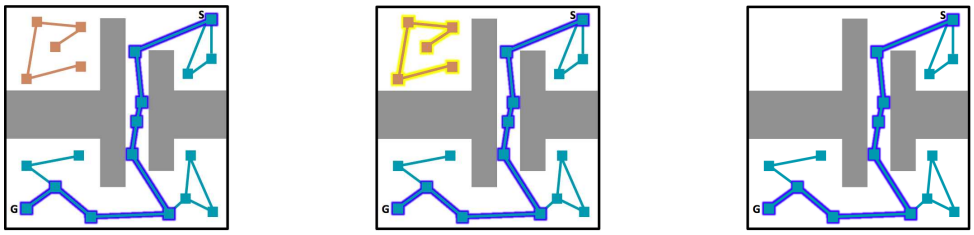
(d) New edge confirmed; shortest path updated

Fig. III.2.: Use of the edge insertion tool to improve the shortest available path.

Vertex/Edge Insertion

The user can add a new node v to the roadmap, thus achieving $V = \{V \cup v\}$. The node is initially placed at the origin of the environment with no incident edges (Figure III.1a), and the user can adjust its configuration as desired using movable sliders and/or typed input for the values of each degree of freedom (Figure III.1b). As the node is adjusted, the user is provided with visual feedback on its feasibility. If the user moves the node to an infeasible configuration, e.g., by colliding it with an obstacle, the color of the node is inverted in warning.

Similarly, the user can add new edge e between a pair of chosen nodes in the map (Figure III.2), resulting in $E = \{E \cup e\}$. Connected components of the graph are updated



(a) Initial roadmap

(b) Group of nodes and edges selected

(c) After deletion of group

Fig. III.3.: Use of the vertex/edge deletion tools to remove an unnecessary connected component.

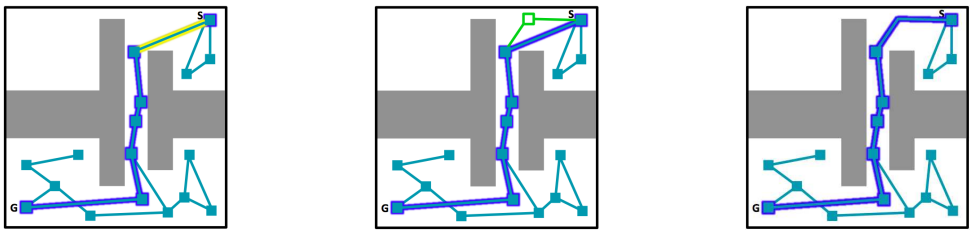
accordingly; for example, a new edge may result in two separate connected components becoming one. One helpful application of the edge insertion utility is the ability to create shortcuts between nodes not initially connected by the planner, which can drastically reduce the final path length (Figure III.2g).

Vertex/Edge Removal

The user can simultaneously delete any number of vertices and/or edges from the roadmap (Figure III.3). The aggregate removal of a group (V', E') of selected vertices and/or edges results in $V = V \setminus V'$ and $E = E \setminus E'$ for the vertex and edge sets of G, respectively. In practice, the user may remove a single node that looks unhelpful, a group of multiple edges at a time to reduce roadmap density, or even an entire connected component. As with vertex and edge insertion, the connectivity of the graph is updated accordingly.

Vertex Adjustment

As outlined in the insertion case, the user can adjust a node's configuration (Figure III.1b), with feasibility checks and resultant visual feedback all the while. The user might, for example, decide to push a node closer to a more interesting area of the environment or attempt to center it in a more desirable position with regard to its incident edges. If the user moves the node in an infeasible way, e.g., out of bounds, the planner responds by inverting



(a) Selection of candidate edge

(b) Adjustment of new intermediate configuration

(c) New intermediate confirmed

Fig. III.4.: Use of the edge adjustment tool to provide more clearance from an obstacle.

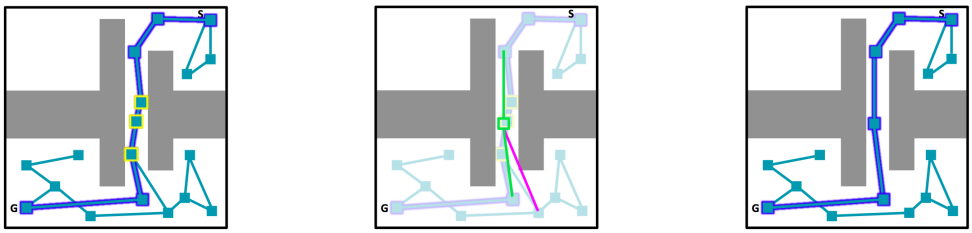
the node's color. Finalizing an invalid position for a vertex is prohibited; however, invalid incident edges are allowed and will be automatically removed.

Edge Adjustment

We define an edge as a polygonal chain through C_{space}; thus, it is not necessarily a straight line and can accommodate a number of *intermediate configurations* along its length. The user can modify a roadmap edge by adding, removing, or adjusting such intermediates (Figure III.4). As in the previous cases, the user's actions are reciprocated by visual feedback, such as validity colors for each intermediate and the disallowment of infeasible changes.

Vertex Merging

Finally, the user can merge a group of vertices into a supervertex (Figure III.5), which can help, e.g., to condense an area of redundant samples. Let S be a subset of nodes in V selected for merging, and consider a function f which maps every vertex in $V \setminus S$ to itself, and every vertex in S to a new supervertex w. Merging the nodes in S results in a new graph $G' = (V', E')$, where $V' = V \setminus S \cup \{w\}$ and $E' = E \setminus \{e | e = (u, v), u, v \in S\}$. Furthermore, for every node $x \in V$, $x' = f(x) \in V'$ is incident to an edge in E' if and only if the corresponding edge in E is incident to x in G. The new supervertex is centered among the nodes selected



(a) Candidate node selection

(b) Adjustment of supervertex

(c) Completed merge, with invalid edges automatically removed

Fig. III.5.: Use of the vertex merging tool to condense an area of samples within a narrow passage. In (b), the map is faded out to make the preview elements more clear.

for its creation (i.e., configured with the average of their DOFs) and can be adjusted, with visual feedback provided as in previous cases.

The Interface

In practice, our map editing interface is presented as a simple, dialog-based extension of [27]. The user might work in a 2D environment similar to that of Figure III.6.

Map operations that involve the direct selection of elements are straightforward. In our interface, they are carried out by mouse clicks and appear much as they do in previous figures (the screenshots in Figure III.7a and b show the expected yellow outlines around a selected node and a selected edge, respectively).

For utilities that involve modifying configurations–namely, the vertex addition and adjustment tools, as well as the vertex merging and edge adjustment tools (for modifying the supervertex and placing intermediate configurations, respectively), the user works with a dialog like that of Figure III.7c. The dialog displays a slider for each degree of freedom, allowing the user to adjust each dimension individually. To the right of the slider, the display box also allows typed entry of the desired value, enabling more granularity in the modification.

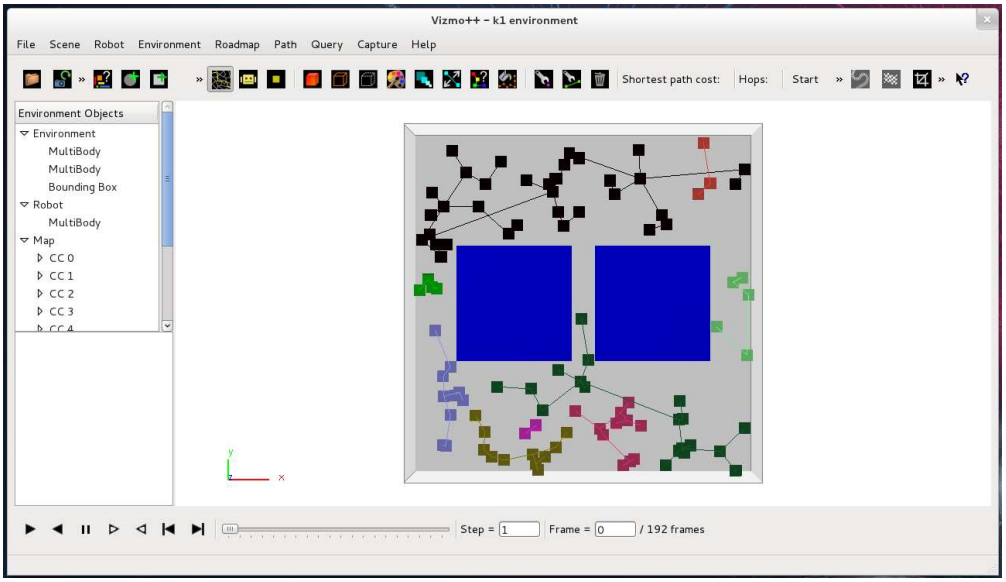
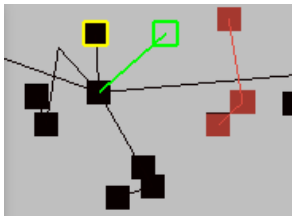
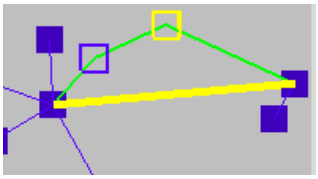


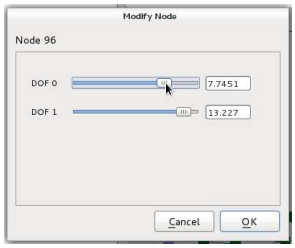
Fig. III.6.: Main window of Vizmo++, the visualization software that incorporates our motion editing tools. A basic environment with two large obstacles and an accompanying roadmap have been loaded.



(a) Vertex adjustment, showing original (selected) and valid preview (green wireframe) configurations



(b) Adjusting an intermediate configuration along an edge



(c) The configuration adjustment dialog

Fig. III.7.: A few screenshots of the real map editing interface.

CHAPTER IV

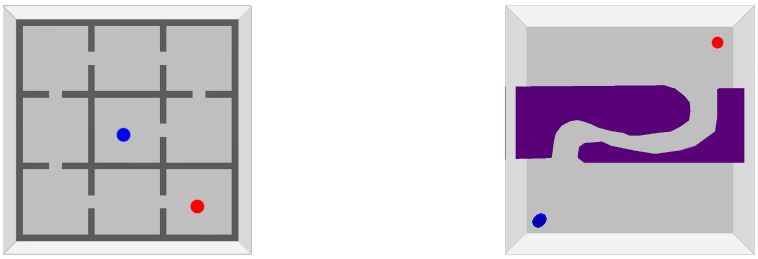
EXPERIMENTAL ANALYSIS

In this section, we evaluate our map editing tools against an automatic graph spanning algorithm [19] for the refinement of dense roadmaps. We show that the tools are promising for some forms of map quality improvement, but more questionable for others, depending on the intuitiveness of the user interface in each situation.

Experimental Setup

All experiments were run on a Dell Optiplex 780 computer running Fedora 17 with an Intel Core 2 Quad CPU 2.83 GHz processor and the GNU gcc compiler version 4.7. We generated dense roadmaps with PRM*, which was implemented in a C++ motion planning library developed in the Parasol Lab at Texas A&M University. This library uses a distributed graph data structure from the Standard Template Adaptive Parallel Library (STAPL) [26], a C++ library designed for parallel computing. Within PRM*, we used a Euclidean distance metric and straight-line local planning. The planner was run until an example query could be solved.

For the automatic refinement tests, the PRM* maps were sparsified with a C++ implementation of the randomized $(2k-1)$ spanner described in [19], with k values of 2, 4, and 6. In the case of our collaborative system, the utilities were implemented as an extension of Vizmo++ [27], a visualization tool for robotic motion planning developed in the Parasol Lab at Texas A&M University. The user, a senior undergraduate student studying motion planning, was presented with a dense PRM* map for each setup and allowed to modify it as much as desired with any of the editing tools, e.g., by adding shortcut edges or merging groups of nodes into supervertices.



(a) Squares (b) Curved Tunnel

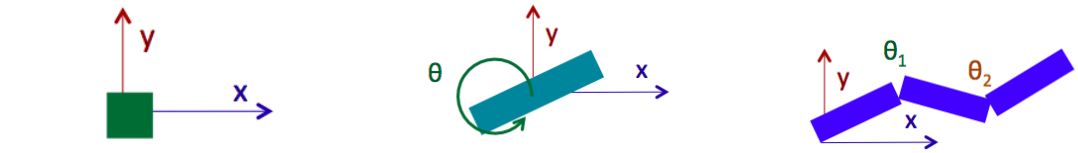
Fig. IV.1.: Environments for experimental analysis. Note that the blue and red dots represent only the locations of the start and goal, not the actual robot and its physical characteristics.

Figure IV.1 shows the 2-dimensional scenarios used in testing. In Squares (Figure IV.1a), a robot must traverse through a series of narrow passages in order to get from the center cell to the one on the bottom right. Three robot types were used in this environment:

- A simple, 2-DOF planar robot with the ability to translate in the x and y directions (Figure IV.2a)
- A planar rectangular robot, with the x and y translation as above plus rotation along the z axis (Figure IV.2b)
- A simple 2D linkage, consisting of three rectangular links connected by revolute joints along the z axis (thus, 4 DOFs total) (Figure IV.2c)

In Curved Tunnel (Figure IV.1b), a robot must traverse through a curved narrow passage between the start and goal configurations. The first two robots (2-DOF and 3-DOF) used in Squares were tested in this environment.

In each case, we analyze the tradeoffs of user vs. spanner post-processing with respect to the time required to carry out the refinements, the numbers of vertices and edges in the graph, and the shortest available path length, as measured by Dijkstra's algorithm. Measurements are shown as the average of 10 randomly-seeded trials, with the graph size and shortest path length metrics normalized against the original PRM* roadmaps.



(a) Planar translational (2-DOF)

(b) Planar rectangular with rotation (3-DOF)

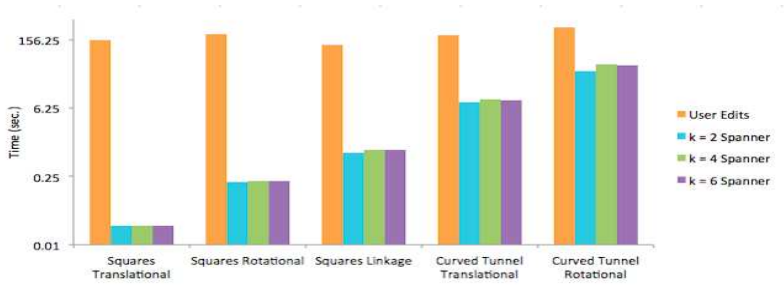
(c) Linkage (4-DOF)

Fig. IV.2.: Robot types used in experiments

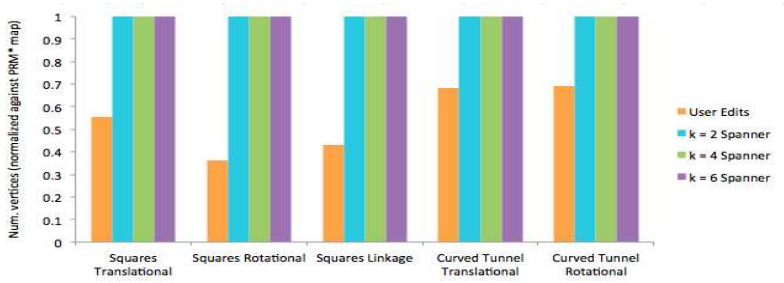
Analysis

Figure IV.3a shows a log scale of the time required for the different map modification processes. This is the clearest weak point for the user editing tools. Times for manual modification with validity feedback were generally on the order of 3 to 4 minutes, while the spanners, especially in simpler cases, were computed on the order of seconds or fractions of seconds. This is hardly surprising, since we have emphasized speed as a central contribution from the automatic side of a cooperative planning system. In a more sophisticated setup–with planner-to-user communication realized as live feedback during map *construction* rather than post-processing–the temporal benefits of human help may become more clear, especially in high-dimensional scenarios.

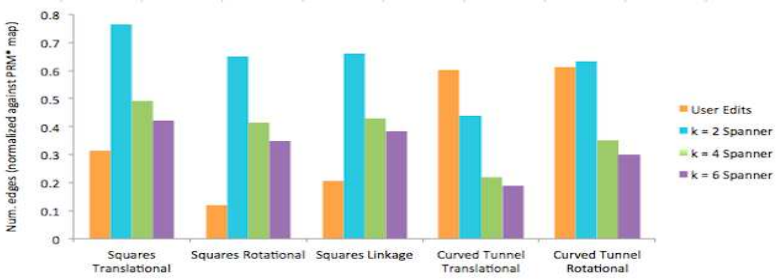
Figure IV.3b compares the numbers of remaining vertices in the roadmaps after post-processing. By definition, the randomized $(2k-1)$ spanner only removes roadmap edges; thus, user editing tools have the clear advantage in this sense. In all scenarios, the user was able to decrease the size of the dense PRM* vertex set by at least 30%, with best performance in the Squares environment, which was simpler for the planner in general and, thus, characterized by smaller input graphs. Interestingly, the user could not remove as many vertices from the Squares Translational environment as from the more complex Squares environments. This is likely because the Squares Translational PRM* maps were the smallest overall–so much that the vertices they did have were more likely to be part of the shortest path and, thus, could not always be safely removed.



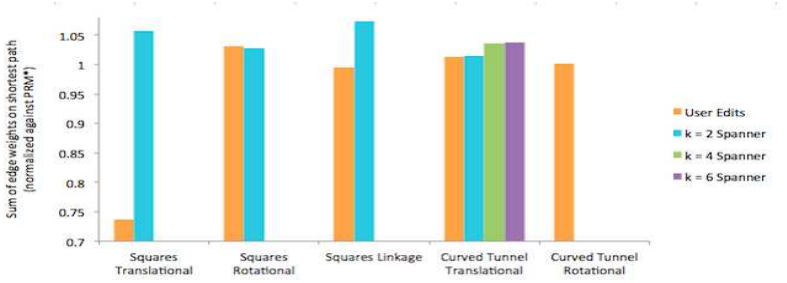
(a) Comparison of time required for map modifications



(b) Number of vertices after modifications



(c) Number of edges after modifications



(d) Shortest path cost after modifications. Absent bars indicate scenarios in which the $(2k-1)$ spanner could not compute a connected graph; thus, a path was not found.

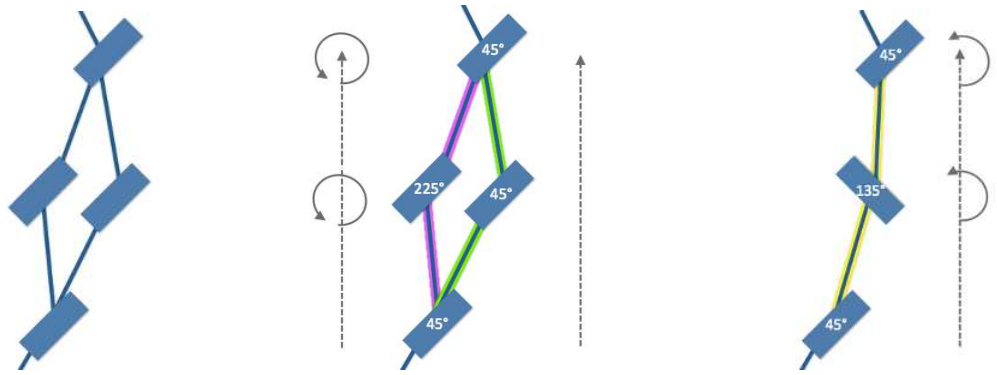
Fig. IV.3.: Comparison of map quality metrics after post-processing.

As shown in Figure IV.3c, the user's ability to outperform the spanner on edge removals varied with the environment. In the Squares scenarios, the user could sparsify the input graphs with ease and often finished with well under 30% of the original edge density. Though the spanners could remove progressively more edges as the k parameter increased, they tended to retain nearly or well over 40% of the edges. In the Curved Tunnel environments, the trend was effectively reversed. The user could remove around 40% of the original edges, but the spanners usually removed far more. This is consistent with the limited rendering capability of our visualization system. As the Curved Tunnel maps had tens of thousands of edges, user modifications were extremely slow and cumbersome and were not carried out to the same extent.

Perhaps the most interesting results were those of the shortest path lengths after map modifications. In Figure IV.3d, it is clear that the user was not always very helpful in reducing–or even preserving–the shortest path length from PRM*. In the Squares Rotational and both Curved Tunnel scenarios, for example, user modifications resulted in slightly longer average path lengths. The spanners were also detrimental, though not to as high of a degree as one might expect.

The edge and path length results confirm the importance of a high-quality user interface for an effective collaboration system. In particular, we see that the user performs best when our map editing tools provide an intuitive, "what you see is what you get" picture of the operations at hand. In the Squares Translational case, for example, it is not surprising that the user could reduce the path length by over 25%, as the cost of, say, a new shortcut edge between close nodes is quite well correlated with the way a user might visually perceive it.

On the other hand, more complex robots are harder to visualize with our interface. In some cases, this can even be costly. Consider a planar robot that is also able to rotate along its z axis. While editing a roadmap, the user might encounter a pair of seemingly redundant nodes (Figure IVa) and decide to merge them into one. Since the true angle of the configurations is not indicated in the interface, it could be that these apparently identical configurations



(a) A misleading scenario (b) Initial path costs (c) After a merge

Fig. IV.4.: A merge operation that is more costly than expected. Dotted gray lines show required types of movement along each edge.

are actually 180°different from one another. As shown in Figures IVb and IVc, merging them together (which averages their configurations) could result in the requirement of a more expensive trajectory (yellow) than the one preferred in the initial case (green), which required no rotational work.

CHAPTER V

CONCLUSION

This work introduced a set of foundational tools for *motion editing*, which we defined as the lowest level of interaction in the *collaborative planning* paradigm. Our interface allows a user to post-process a planning graph by modifying and removing vertices and edges as desired, with reciprocal validity feedback from the planner. In experiments against a graph spanning algorithm, we found that user interaction could significantly reduce the vertex and edge density of a roadmap as compared to the spanner. Path length improvements, though promising in low-dimensional problems, were limited by the user interface in more complex environments. Future work should focus on the improvement of this interface, particularly in making intricate scenarios more digestible to the human eye. This will require innovative ways to re-structure high-dimensional problems, e.g., through clever mappings into smaller variable spaces or carefully-constructed visual and haptic cues. We believe that human intuition–with its unrivaled flexibility and "big-picture" understanding–can offer substantial benefits to cooperative motion planning, among other applications, though its unusual power demands simplicity to be most effectively exploited.

REFERENCES

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. OBPRM: an obstacle-based PRM for 3d workspaces. In *Proceedings of the third workshop on the algorithmic foundations of robotics on Robotics : the algorithmic perspective: the algorithmic perspective*, WAFR '98, pages 155–168, Natick, MA, USA, 1998. A. K. Peters, Ltd.

[2] O. B. Bayazit, G. Song, and N. M. Amato. Enhancing randomized motion planners: Exploring with haptic hints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 529–536, 2000.

[3] O. B. Bayazit, G. Song, and N. M. Amato. Enhancing randomized motion planners: Exploring with haptic hints. *Autonomous Robots, Special Issue on Personal Robotics*, 10(2):163–174, 2001. Preliminary version appeared in *ICRA 2000*, pp. 529–536.

[4] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1018–1023, May 1999.

[5] J. Denny and N. M. Amato. Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 297–312, Cambridge, Massachusetts, USA, June 2012.

[6] A. Dobson and K. E. Bekris. Improving sparse roadmap spanners. In *IEEE International Conference on Robotics and Automation (ICRA)*, 05/2013 2013.

[7] A. Dobson, A. Krontiris, and K. Bekris. Sparse roadmap spanners. In E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus, editors, *Algorithmic Foundations of Robotics X*, volume 86 of *Springer Tracts in Advanced Robotics*, pages 279–296. Springer Berlin Heidelberg, 2013.

[8] C. Guo, T. Tarn, N. Xi, and A. Bejczy. Fusion of human and machine intelligence for telerobotic systems. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3110–3115, 1995.

[9] D. Hsu, T. Jiang, J. Reif, and Z. Sun. Bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4420–4426, 2003.

[10] Y. Hwang, K. Cho, S. Lee, S. Park, and S. Kang. Human computer cooperation in interactive motion planning. In *Proc. IEEE Int. Conf. Adv. Robot. (ICAR)*, pages 571–576, 1997.

[11] I. Ivanisevic and V. Lumelsky. Human augmentation in teleoperation of arm manipulators in an environment with obstacles. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1994–1999, 2000.

[12] I. Ivanisevic and V. Lumelsky. Augmenting human performance in motion planning tasks-the configuration space approach. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2649–2654, 2001.

[13] I. Ivanisevic and V. J. Lumelsky. Configuration space as a means for augmenting human performance in teleoperation tasks. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 30(3):471–484, Jun 2000.

[14] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *CoRR*, abs/1105.1186, 2011.

[15] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.

[16] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *Int. J. Robot. Res.*, 20(5):378–400, May 2001.

[17] S. Lee, G. Sukhatme, G. J. Kim, and C.-M. Park. Haptic teleoperation of a mobile robot: A user study. *Presence: Teleoperators and Virtual Environments*, 14(3):345–365, 2005.

[18] J.-M. Lien and E. Pratt. Interactive planning for shepherd motion, March 2009. the AAAI Spring Symposium.

[19] J. D. Marble and K. Bekris. Computing spanners of asymptotically optimal probabilistic roadmaps. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4292–4298, Sept 2011.

[20] J. D. Marble and K. E. Bekris. Asymptotically near-optimal is good enough for motion planning. In *The International Symposium on Robotics Research (ISRR)*, 2011.

[21] S. Parikh, V. G. Jr., V. Kumar, and J. O. Jr. Incorporating user inputs in motion planning for a smart wheelchair. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2043–2048, 2004.

[22] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, October 1979.

[23] D. Shaharabani, O. Salzman, P. K. Agarwal, and D. Halperin. Sparsification of motion-planning roadmaps by edge contraction. *CoRR*, abs/1209.4463, 2012.

[24] A. P. Singh, J.-C. Latombe, and D. L. Brutlag. A motion planning approach to flexible ligand binding. In *Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 252–261, 1999.

[25] M. Taïx, D. Flavigné, and E. Ferré. Human interaction with motion planning algorithm. *Journal of Intelligent & Robotic Systems*, 67(3-4):285–306, 2012.

[26] G. Tanase, A. Buss, A. Fidel, Harshvardhan, I. Papadopoulos, O. Pearce, T. Smith, N. Thomas, X. Xu, N. Mourad, J. Vu, M. Bianco, N. M. Amato, and L. Rauchwerger. The stapl parallel container framework. In *PPOPP*, 2011.

[27] A. Vargas Estrada, J.-M. Lien, and N. M. Amato. Vizmo++: a visualization, authoring, and educational tool for motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 727–732, May 2006.

[28] W. Wang, D. Balkcom, and A. Chakrabarti. A Fast Streaming Spanner Algorithm for Incrementally Constructing Sparse Roadmaps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, 2013.

[29] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1024–1031, 1999.

[30] Y. Yan, E. Poirson, and F. Bennis. Integrating user to minimize assembly path planning time in plm. In A. Bernard, L. Rivest, and D. Dutta, editors, *Product Lifecycle Management for Society*, volume 409 of *IFIP Advances in Information and Communication Technology*, pages 471–480. Springer Berlin Heidelberg, 2013.

[31] Y. Yang and O. Brock. Adapting the sampling distribution in prm planners based on an approximated medial axis. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 5, pages 4405–4410, 2004.